PIC12C508 Stepper Motor Controller

C. Tavernier

www.tavernier-c.com

When we're not using a stepper motor to ensure precise positioning of a robot element, it can be used as a traction motor, in place of the standard modified servos presented elsewhere in this issue. Under these conditions, there's no longer any need to 'count the steps' the motor has to make, as all we want is to make it rotate continuously in one direction or the other.

Several solutions are open to us for driving the motor, a number of which are presented in this issue: using a specialized stepper motor driver IC, using one or more suitably-programmed microcontroller parallel ports, or building a driver based around conventional logic ICs.

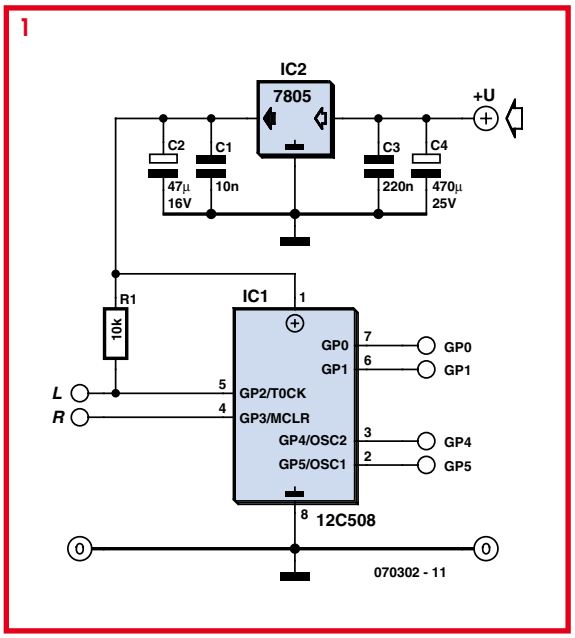
However, these solutions are far from satisfactory when using a stepper motor for traction. They all require pulses to be generated continuously for as long as we want the motor to run, either requiring an additional programmable oscillator, or using up resources from the robot's main microcontroller.

So we've decided to suggest another approach with this stepper motor driver specifically designed for making the motor turn in one direction or another, under the control of a simple logic level. And as the propulsion motors in robots usually go in pairs, we're even going to offer a dual driver, by diverting a very common and inexpensive IC from its original function.

Since a stepper motor used for propulsion doesn't need to be accurate in terms of positioning, and hence, in the precision of the steps, simple single-pole models are eminently suitable. So, our circuit is designed for motors of this type.

This lets us control the motor via two TTL- or CMOS-compatible logic inputs. When these two inputs, labelled L and R, are logic high or floating (they have their own pull-up resistors), the motor stays still, but in braked mode, since it's a stepper motor. When the L input is taken to logic low, the motor rotates in one direction (arbitrarily, to the left, whence the label L) while if the R input is taken low, it turns the other way. If both inputs are taken to ground at the same time, the R input has priority, and so the motor turns in that direction.

The motor's speed of rotation is fixed, but, since we are giving you the source listing of the software used for this application, it's very easy for you to modify this if it doesn't suit you, or indeed even to include the possibility of external adjustment if necessary.



The circuit of the 'intelligent' part of our controller is shown in **Figure 1,** as you can see it uses a PIC12C508 microcontroller from Microchip. Used here in internal clock and reset circuit mode, it needs no external components for these functions, so all its port lines are available.

Parallel ports GP2 and GP3 are used as inputs, and as GP2 does not have an internal pull-up resistor, this is performed by R1. Parallel ports GP0, GP1, GP4, and GP5 are used as outputs for generating the pulses for the motor windings. These can be amplified by two types of power stages, depending on the type and number of motors to be driven; we'll take a look at those circuits in a moment.

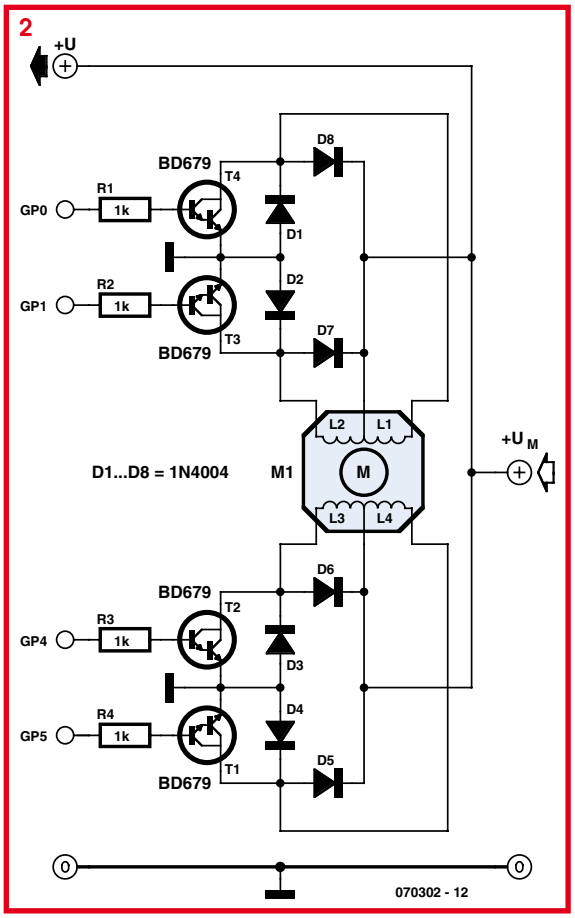
The 12C508 needs to be powered from 5 V, derived from the motor supply by means of a conventional 3-terminal voltage regulator IC2.

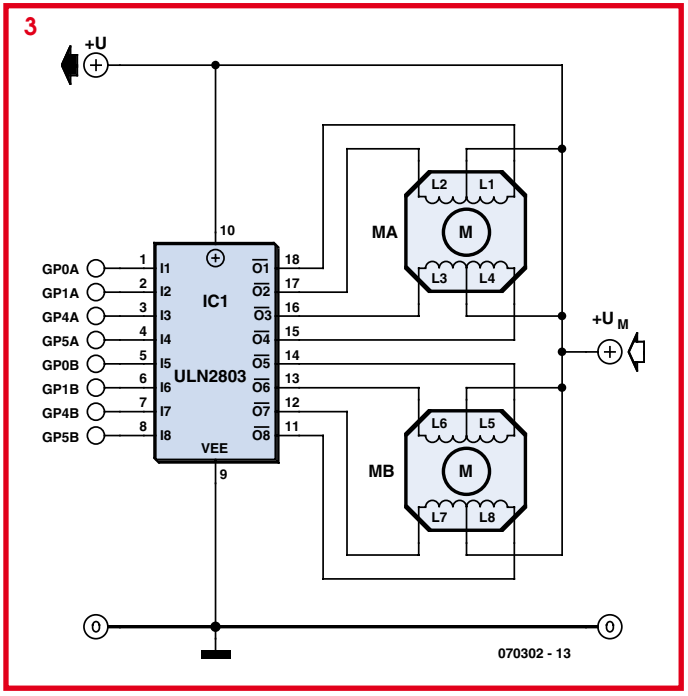
If the controller is only intended for a single motor, or if the motor to be driven draws more than 500 mA per winding, the power stage shown in **Figure 2** can be used. It employs conventional bipolar transistors that, given their characteristics, are able to switch currents of 3 A. Diodes D1–D8 clip the spurious spikes generated by the abrupt switching of the current in the motor windings and protect the transistors.

However, if the motor used draws less than 500 mA, and more importantly, if you need to drive two motors of this type, an elegant and ingenious solution exists, as shown in **Figure 3**. This uses a standard ULN2803, usually used to drive relays, but which includes eight medium-power Darlingtons along with their protection diodes. So, this IC is able to properly drive any kind of single-pole stepper motor, as long as the voltage required doesn't exceed 50 V and the current per winding is under 500 mA.

In addition, as the ULN2803 contains eight identical stages, it can be preceded by two controllers like the one in Figure 1 and in this way drive two robot propulsion motors: one on the left and one on the right, marked MA and MB in this figure.

Constructing one or other of these versions is very straightforward. The PIC 12C508 needs to be programmed with the file that you'll find in object





small aluminium plate a few cm². To simplify mechanical construction, it can be common to the four transistors, but in this case you'll need to use the standard insulating accessories of mica washers and shouldered washers, as the collectors of these transistors are connected to the metal parts of their cases.

If you construct the ULN2803-based version, there are no special precautions to be observed, other than to not exceed the IC's maximum current capacity of 500 mA.

As we are providing you with the full source listing of the software programmed into the 12C508, you'll be able to modify it to suit your needs. If you are unfamiliar with PIC microcontroller assembler, here are the details you'll need for the most important modification you might want to make: changing the speed of the control pulses to the motors, and thus, their speed of rotation. The control word may be found in **Table 1**.

To do this, all you have to do is modify the binary constant on the line:

```
MOVLW B'10010101'
```

just above the line containing OPTION in the source listing. With the original value, the duration of one step is 8 ms, but the table above indicates what constant to use according to the step duration that you may want.

(070302-I)

form, as well as in source form, in case you'd like to modify it, on the Elektor website, as well as on the author's own site (www.tavernier-c.com).

If you build the transistor power amplifier, note that T1–T4 don't need a heatsink as long as the motor consumption doesn't exceed 1 A. Otherwise, bolt them onto a

Table 1. Programming step duration by modifying a constant used in the program.

Binary Step constant	duration
10010010	1 ms
10010011	2 ms
10010100	4 ms
10010101	8 ms
10010110	16 ms
10010111	32 ms